Overview of the High Level Architecture and Its Potential for Use in NASA Projects

Michael Reid

Computer Sciences Corporation

Mike.Reid@gsfc.nasa.gov

February 23, 2000

What is the HLA?

- The High Level Architecture (HLA) is a standard framework that supports modeling and simulation.
- "The HLA is the glue that allows you to combine computer simulations into a larger simulation."[1]

1. F. Kuhl, R. Weatherly, J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, p. 1, Prentice Hall PTR, 1999.

Origins of the HLA

- The HLA was developed by the DoD's Defense Modeling and Simulations Office (DMSO).
- It was designed to meet the DoD's needs in the modeling of large-scale systems such as war games, weapons systems, and training.

Purpose of the HLA

- Facilitates the reuse of components
- Defines a common interface for individual components
- Provides a framework for creating simulation systems from a collection of individual, separate distributed applications

Adoption of the HLA

- The DoD has mandated that all of its future simulators must be HLA-compliant.
- An upcoming industry standard (IEEE-1516).
- Being adopted by other NATO defense organizations.
- Expanding beyond the defense arena:
  - Industrial process simulations
  - Weather system modeling
  - Games

What the HLA Can Bring To NASA

- Many of NASA's simulation and modeling needs are similar to those of the DoD.
- NASA can take advantage of R&D already completed and paid for by the DoD.
- Standardization reduces costs.
- Potential for reuse of DoD and other IEEE–1516 compliant simulators.
- Facilitates collaboration with the DoD.

What the HLA Can Bring To NASA (continued¼)

- Fully supports distributed computing
- Supports collaborative development over different sites
- Integrates applications running on different platforms
- Scales up
- The future of simulations and modeling??

ISE Potential?

- NASA's *Intelligent Synthesis Environment* (ISE) Initiative seeks:†
  - Rapid Synthesis and Simulation Tools
  - Cost and Risk Management Technology
  - Life-Cycle Integration and Validation
  - Collaborative Engineering Environment
- Simulation, modeling, virtual environments, shared product models

†National Research Council, *Advanced Engineering Environments: Achieving the Vision, Phase 1*, pp. 14–16, National Academy Press, Washington, D.C. 1999. [ISBN 0-309-06541-0].

HLA Terminology

- Federate
  - An individual simulator application.
- Federation
  - A simulation composed of two or more (often many more) federates integrated together.
- Federation Execution
  - A session in which a federation is running.

Components of the HLA

- A specification for an overall architecture and a set of interfaces.
- A set of rules governing federate and federation design.
- A standard application programming interface (API).
- A Runtime Infrastructure (RTI) software package, which implements the API.

Elements of a Federation

- A Federation Object Model (FOM)
  - A common object model for data exchange between federates running in the federation.
- The Runtime Infrastructure (RTI).
- Two or more HLA-compliant federates running within the federation execution.

Elements of a Federate

- A Simulation Object Model (SOM)
  - Defines the data objects shared with other federates.
- A binding to the RTI library (`libRTI`).
  - Allows the federate to communicate with the RTI.
- An implementation of the FederateAmbassador.
  - A set of callbacks implemented by the given application, which allow the RTI to send information and directives to the federate.

Communications

- Federates communicate within the federation exclusively through the RTI.
- Means of sharing data
  - Published data objects
    - Essentially, data containers, which can be updated by the owning federate, are visible to other federates, and persist within the federation execution until deliberately destroyed.
  - Interactions
    - Broadcast messages, which can be received by other federates.
    - Interactions do not persist after being sent.

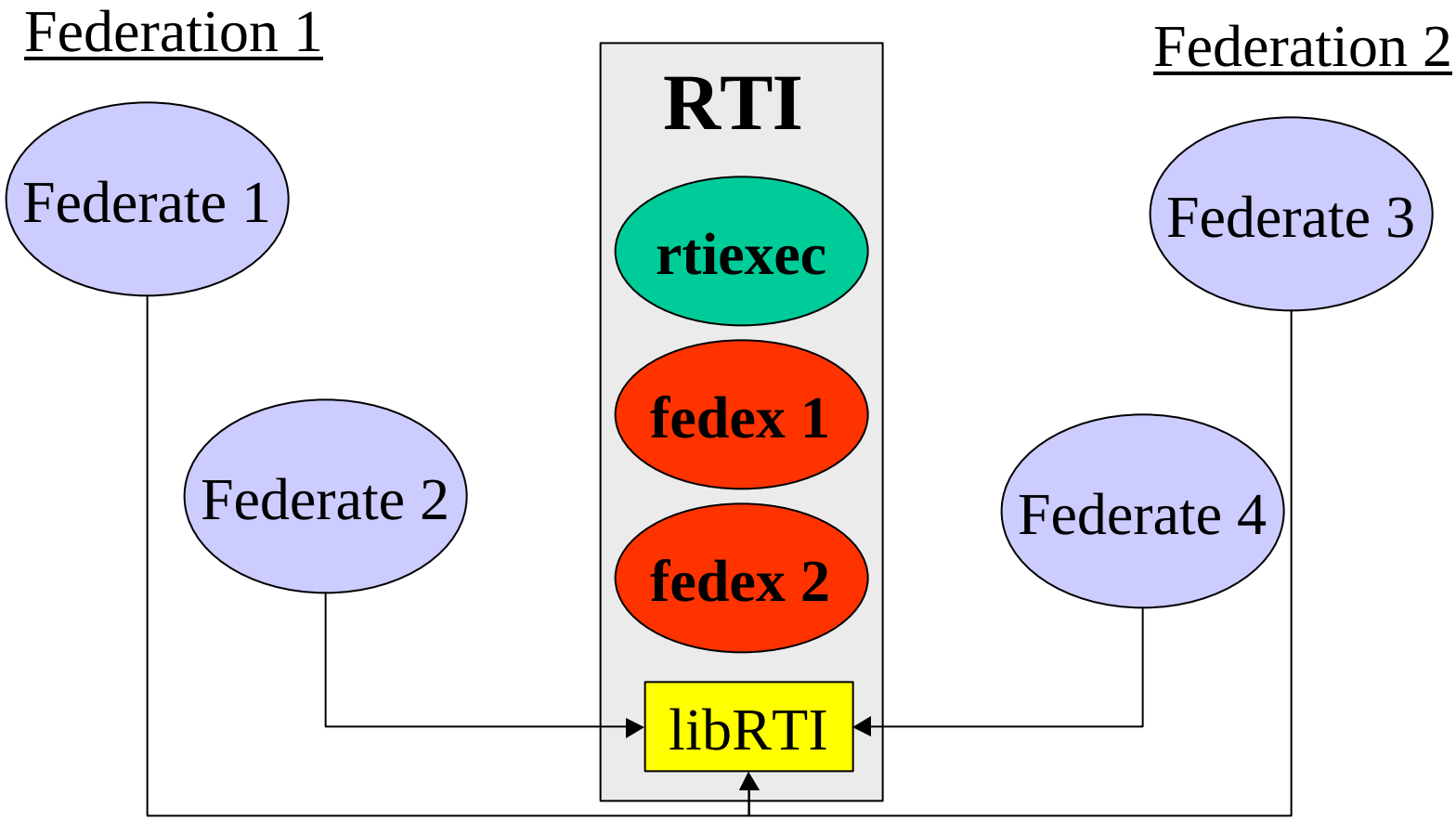
What the RTI Does

- Integrates the component federates into a running simulation.
- Handles all communications between federates.
- Synchronizes the federation.
- Controls data distribution and visibility within the federation.

Components of the RTI

- The `libRTI` library.
  - Implements the HLA API.
- The *rtiexec* daemon.
  - Directs federates to the correct federation execution.
  - Can run more than one federation.
- The *fedex* daemon.
  - Manages the federation.
  - Integrates the federates within the federation execution.
  - Handles communications between federates.

Federation Execution



RTI Initialization Files

- The Runtime Initialization Data (RID) file.
  - Defines runtime parameters.
  - Specifies the network interfaces and protocols.
  - Sets RTI system configuration parameters.
- The Federation Execution Data (FED) file.
  - One per federation execution.
  - Defines the object and interaction classes, which are shared within the federation.

The DMSO RTI

- Available for FREE from DMSO.
- Supported platforms:
  - Most major commercial UNIX® platforms.
  - Linux™ (for Intel-based systems).
  - Windows NT®
  - VxWorks®
- Language bindings:
  - C++, Java, Ada, CORBA IDL

Services Provided by the RTI

- Federation Management
- Declaration Management
- Object Management
- Ownership Management
- Time Management
- Data Distribution Management

Federation Management

- Incorporates the Federation Execution Data (FED) configuration file (the *federation*.fed file)
  - Derived from the FOM
  - Defines the data objects and interactions shared between federates
- Creates federations
- Joins federates to the federation
- Sets federation-wide synchronization points
- Effects saves and restores
- Resigns federates from the federation
- Destroys federations

Declaration Management

- Publication
  - Federates “publish” through the RTI information that they wish to share with other federates
- Subscription
  - Federates “subscribe” to information that they wish to receive from other federates
- The RTI handles the communications

Object Management

- Provides for the exchange of data among federates
  - Registration of new data objects
  - Updating data objects
  - Sending and receiving interactions
- Informs other, interested federates about new or updated data objects and new interactions

Ownership Management

- Keeps track of which federates own which objects
  - Federates “own” instances of data objects or attributes within an instance of a data object
- Makes sure that only the owning federate may update an objects attribute
- Transfers ownership between federates
- Makes sure that only the owning federate may destroy an object

Time Management

- Synchronizes the federation
- Controls when, in logical simulation time, a given federate receives interactions or notices of updates to the attributes of subscribed data objects
- Controls when federates can advance their internal clocks
- Provides for both event-driven and clock-driven simulations

Time Management (continued)

- Federates are declared to be:
  - Time Regulating
    - Federate can set the pace of the advance of logical time within the federation execution
  - Time Constrained
    - Federate can be throttled by time regulating federates
  - Both time regulating and constrained
  - Neither time regulating nor constrained

Data Distribution Management (DDM)

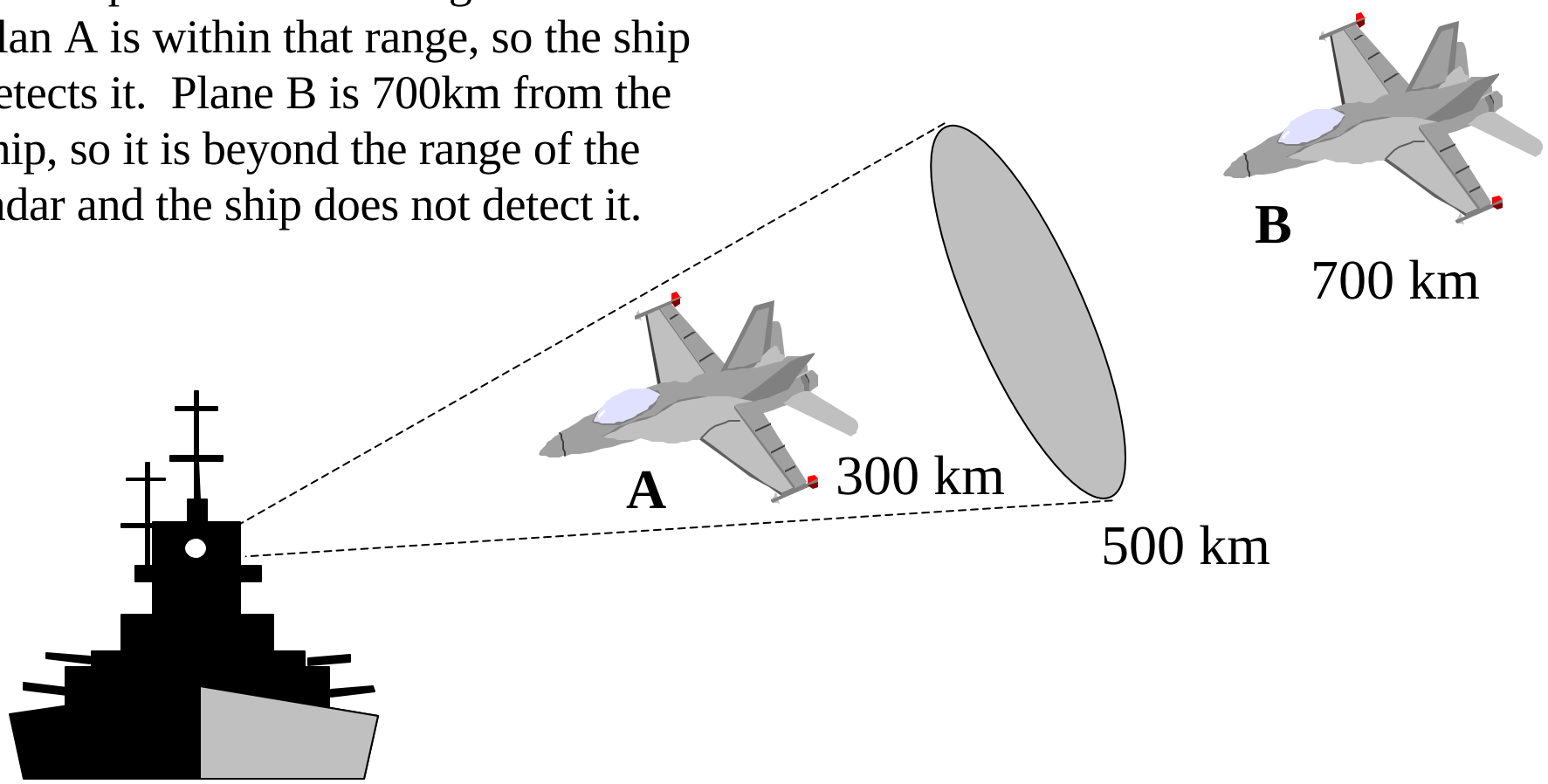
- Defines “routing spaces,” which define the ranges of data distribution
- Allows federates to place conditions on whether or not they receive interactions or notifications of updates to object attributes
- Useful for defining the scope of visibility of data

DDM Example

- An engagement between a surface warship and two attacking aircraft is being simulated:
  - The ship simulator contains a radar and a anti-aircraft missile system
  - The shipboard radar federate defines a routing space based upon distance from the ship
  - The aircraft federate publishes aircraft objects through the RTI, which contain the locations of the aircraft
  - The RTI will inform the radar federate of the aircraft objects only if their locations fall within the radar's routing space

DDM Example (continued¼)

The ship's radar has a range of 500km. Plan A is within that range, so the ship detects it. Plane B is 700km from the ship, so it is beyond the range of the radar and the ship does not detect it.



Potential NASA Applications

- Spacecraft simulation
- Ground station simulation
- Natural Object Modeling
  - Earth Science
  - Astronomy
- Overall Space Missions

Spacecraft Simulations

- Spacecraft are collections of integrated components.
- Simulators which model-specific components could be integrated into a federation to model the spacecraft.
- Simulated spacecraft components could be used for multiple missions.

Ground Station Simulation

- Ground stations are collections of integrated systems
- Aspects of data capture, commanding and control, and science data processing can be modeled and “play” together in HLA-based simulations along with real components
- Would facilitate early integration and testing

Earth Sciences

- The Earth is an integrated natural system
  - Land masses, hydrosphere, cryosphere, biosphere, atmosphere, magnetosphere, etc.
  - Applications which model particular Earth systems could be integrated together using the HLA
- Data from Earth sciences spacecraft could be used in the simulations

Astronomy

- HLA-compliant simulators could model astronomical objects
  - Could use data from spacecraft and ground-based observatories
- Simulated Planets and spacecraft could "play" together in simulations
- Distant astronomical objects could be modeled as well

Overall Space Missions

- Objection: So what? Computer modeling of spacecraft and natural objects is nothing new. This has been done for years. Why do we need the HLA?
- Answer:
  - Individual, stand-alone simulators can be integrated to form larger simulations
  - HLA-compliance facilitates reuse

Space Mission Example

- Hypothetical Earth Science Mission:
  - Telemetry generation
  - On-board instrument data generation
  - Orbit calculations (probably using COTS)
  - Earth image generation in multiple bands
  - Ground station receivers
  - Ground station science data processing
    - Generation of data products

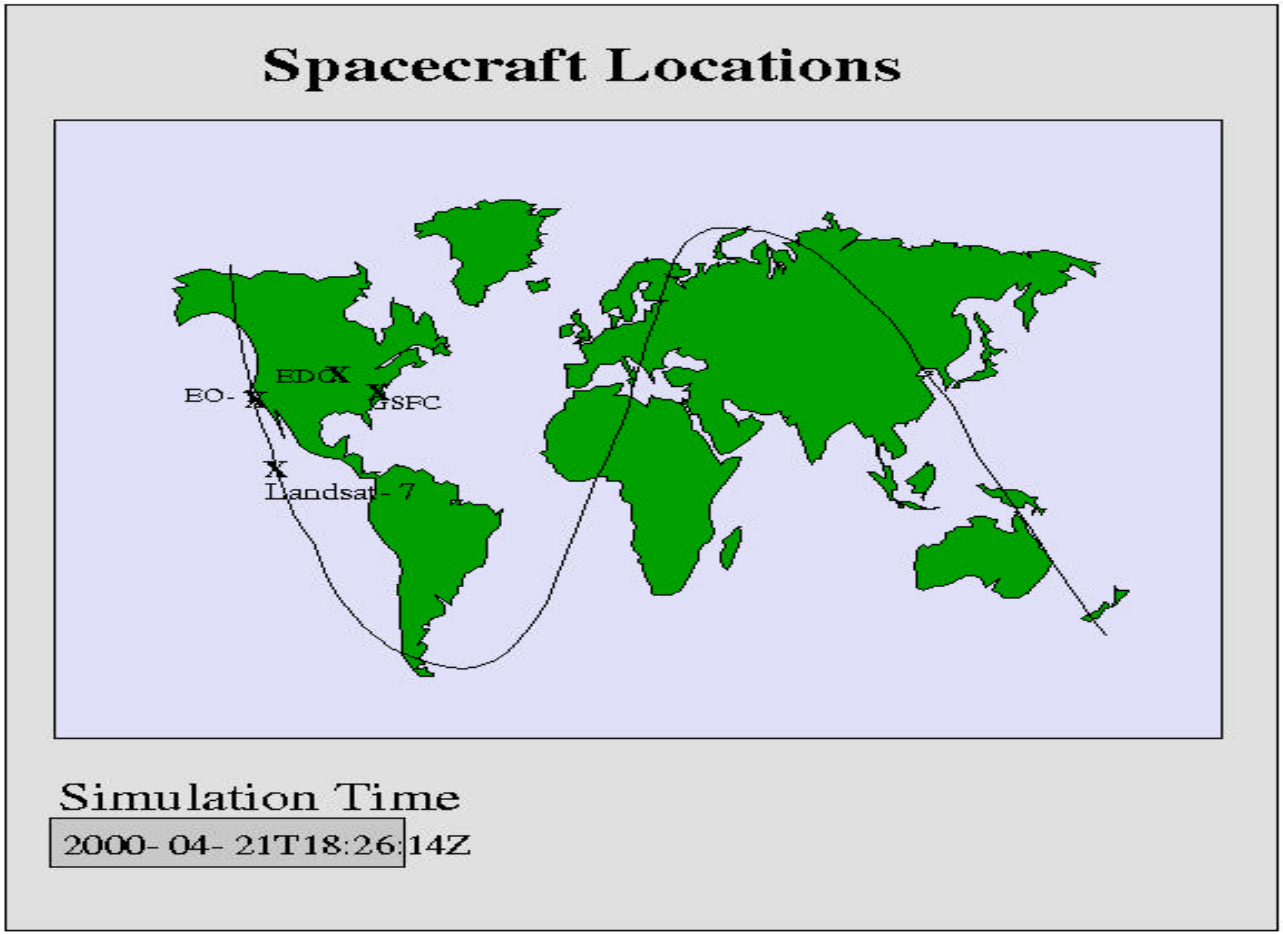
Constellation Flying

- HLA allows unlimited replication of federates.
- A federate which simulates a spacecraft could be duplicated any number of times.
- HLA objects are extensible using inheritance.
- Numerous, similar spacecraft could be added to the same simulation.

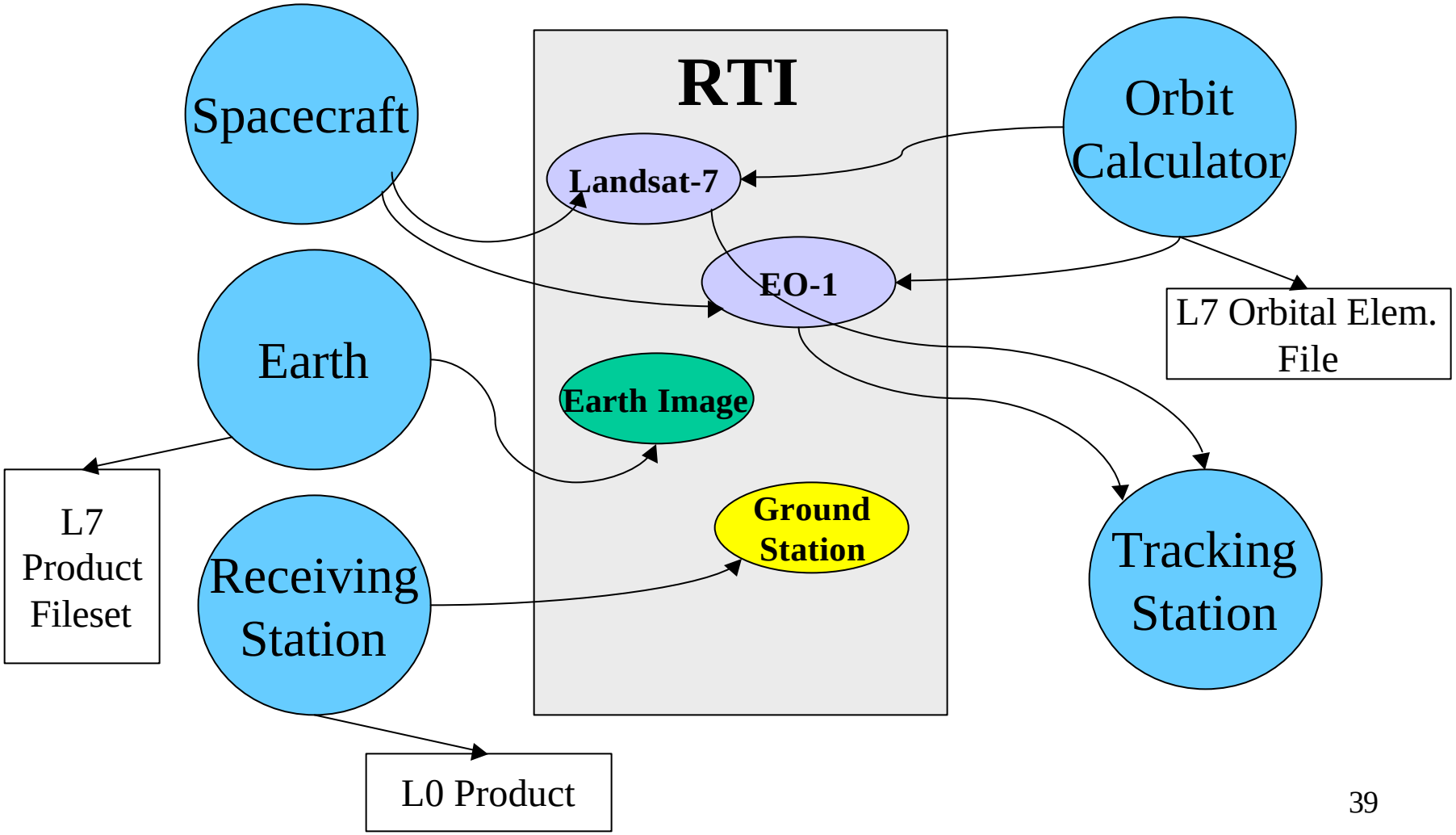
Our R&D Project

- Develop a prototype, HLA-compliant space mission simulator
  - Two spacecraft flying in constellation in Earth orbit
  - Loosely patterned after Landsat-7 and EO-1
- Federates:
  - Spacecraft simulator
  - Earth simulator
  - Orbit calculator
  - Ground station simulator
  - Spacecraft tracking station simulator

Landsat-7 and EO-1



R&D Prototype



The Prototype Demonstrates:

- The viability of the HLA technology for NASA simulations
- The use of already collected science data in mission simulations
  - Landsat-7 science and orbital elements data
- The modular and distributed design concept
- The incorporation of existing software into an HLA-based simulation
  - Orbit calculation programs obtained from Flight Dynamics

More on the HLA

- Websites
  - http://hla.dmso.mil
  - http://www.ecst.csuchico.edu/~hla/
- Articles
  - Dahmann, J., Calvin, J., Weatherly, M., A reusable architecture for simulations. *Communications of the ACM*, 24, 9 (Sep. 1999).
- Books
  - F. Kuhl, R. Weatherly, J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Prentice Hall PTR, 1999. [ISBN 0-13-022511-8].